AI 응용시스템의 이해와 구축

2강. ML 파이프라인, 모니터링, 데이터

출석

대면 Set Up

Final Project: Engineering Design

Final Project

- 과제물: Engineering Design Doc
  - 팀 멤버들 공통 작성
  - Template + 사례 공지 예정.
  - **구글닥에 코멘트**를 활용하여 디자인 토론 + 피드백
  - 학기 중간에 중간 점검 (중간고사 직후)
- 과제물 제출: 14강 수업 직전.
  - 구글닥 코멘트 + 버전 히스토리
  - Bonus Point: 실제 시스템 구축
- 발표: 받은 코멘트를 수렴하여 마지막

한 팀당 3~4명. 전체 5팀.

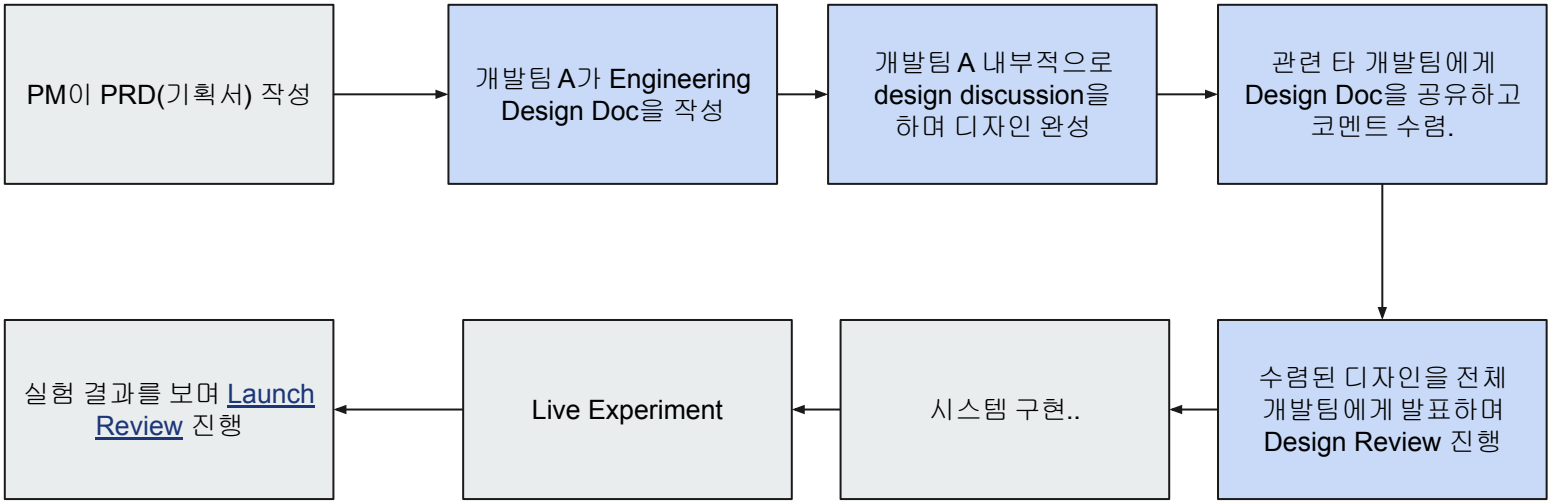
14강 수업 직전에 제출 후 일주일동안 다른 수강생들에게도 코멘트 + 질답 + 제안 수렴.

5팀을 2시간 안에 마치려면 빠듯하니 2팀은 일주일 전 (14강)에 발표 가능?

Final Project: Eng Design + Presentation

- 프로젝트의 목표: **실제 프로덕션**에서 ML모델을 배포할 **디자인**을 직접 진행해 보는
- 테크 기업들 (실리콘밸리 기업들 및 국내 테크기업 포함)에서 사용하는 포맷으로 진

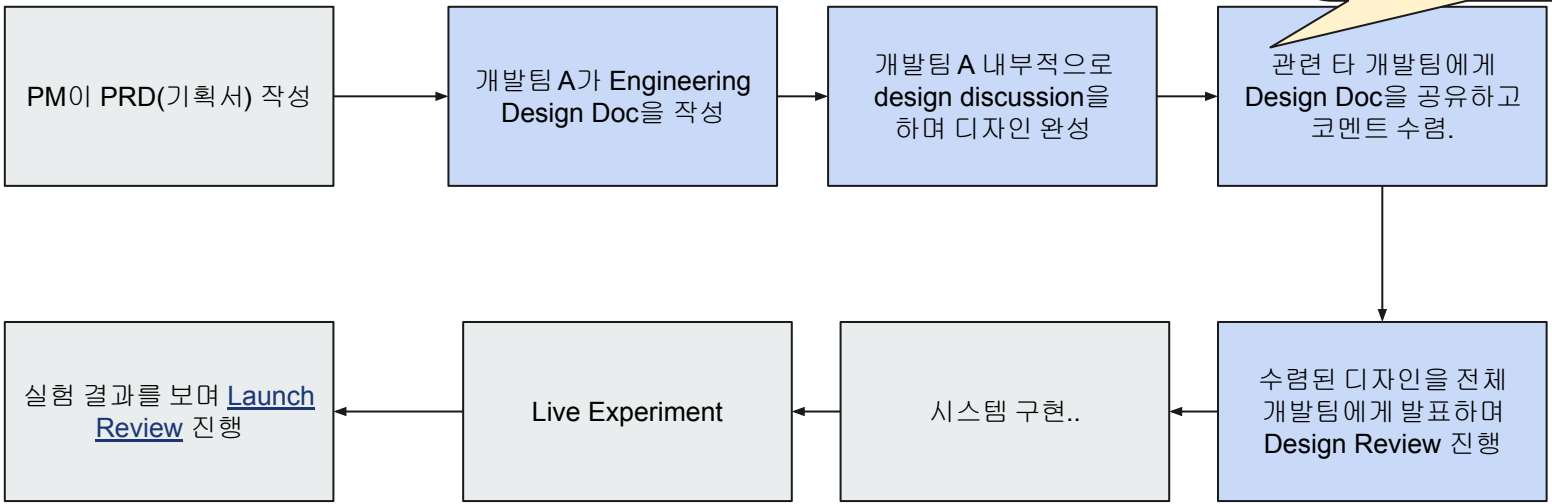
파란 부분을 Final Project로 진행합니다!



Final Project: Eng Design + Presentation

- 프로젝트의 목표: **실제 프로덕션**에서 ML모델을 배포할 **디자인**을 직접
- 테크 기업들 (실리콘밸리 기업들 및 국내 테크기업 포함)에서 사용하는

어떤 코멘트들이 유용한가?
- 사용가능한 다른 솔루션이 기존에 있을 경우 (e.g. Speech2Text)
- 서버 vs edge에 모델 배포
- 테스트는 어떻게 할지?
- 모델 업데이트는?

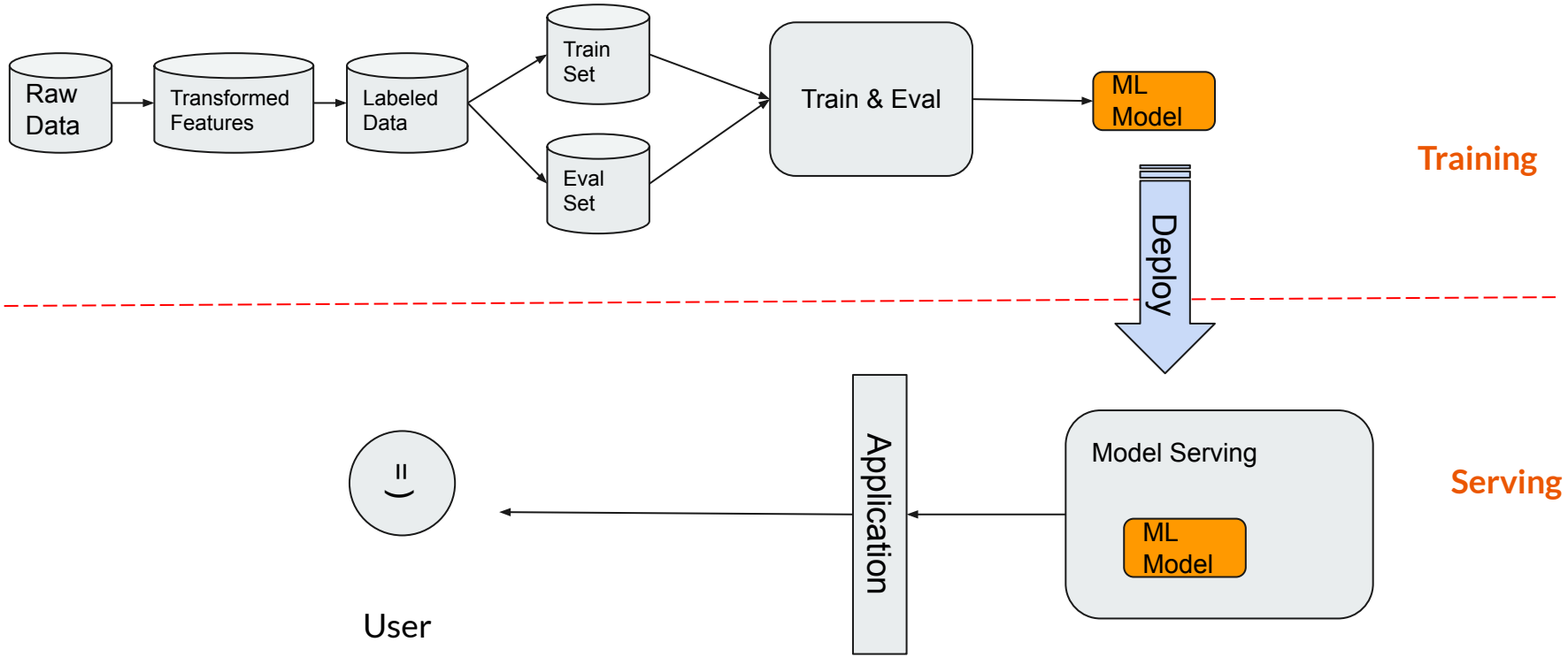


Engineering Design Doc Template

Template Link: “사본만들기"를 하여 팀별로 한 카피 만들어서 진행 하시면 됩니다.

ML Pipeline

Production ML Pipeline (Train & Serving)



프로덕션 ML이 왜 어려운가?

- ML시스템이 **다른 시스템 컴포넌트들 과 연계** 되어야 함.
  - app / frontend: 버전 콘트롤, on-device 모델 업데이트 주기
  - backend infra availability: 학습때 사용한 data source가 추론에도 늘 존재하는가?

- 리소스 최적화 (Compute, I/O, Storage) 필요
  - 모델 학습 때 필요한 리소스 (Compute, I/O)
  - 모델 추론 때 필요한 리소스 및 **Scale Up** (사용자 증가)

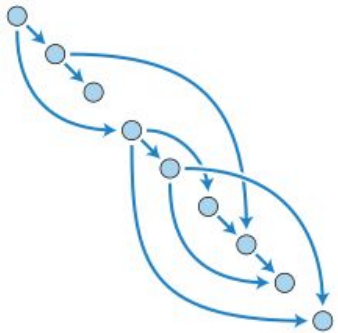
- ML 파이프라인이 다운타임 없이 **Continuous하게 동작**해야 함.
  - Training pipeline: 모델 업데이트할때마다 필요
  - Serving pipeline: 당연히 늘 available

- **지속적으로 변하는 데이터**를 핸들(Data Versioning):
  - Train / Serving Skew를 발견하려면 학습때 사용된 데이터셋과 현재 추론에 쓰이는 데이터셋을 모두 갖고 있어야 함.
  - 데이터 자체가 변한다면?

```
SELECT User_Profile FROM Users
        WHERE User_ID = xxxx AND
                Users.Timestamp = yyyyyyy
```

Structure of ML Pipelines

ML 파이프라인은 구조적으로 **Directed Acyclic Graphs (DAGs)**를 형성합니다.

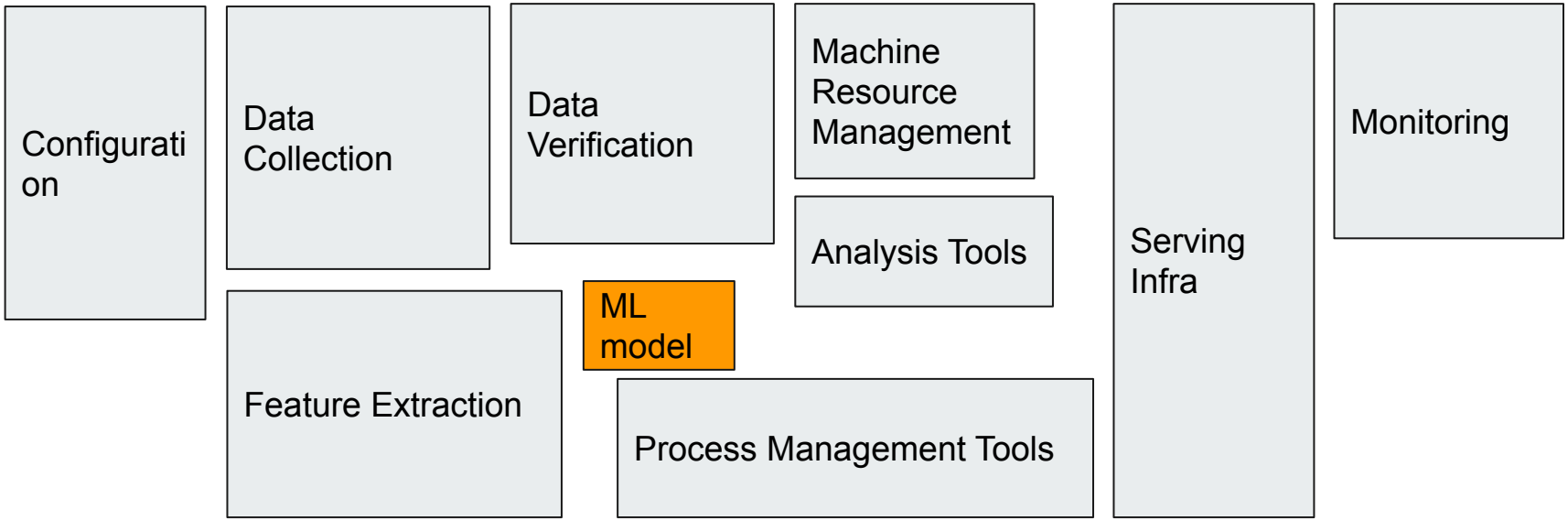


DAG 란?	• 그래프 중 모든 엣지에 방향성(directed edge)가 있지만, 방향 순환 (directed cycle)이 없는 그래프를 뜻함. • 그래프가 DAG일 경우, 각 노드들을 일렬로 정렬하여 엣지들이 한 방향으로 흘러가게 할 수 있다. (Topological Sort)
왜 DAG 구조인가?	• ML 파이프라인에 있는 task들을 순차적으로 + **병렬처리**를 사용하여 스케줄링 할수 있음.

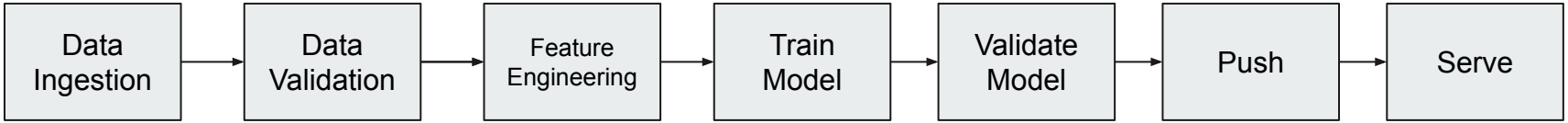
이렇게 task들을 스케줄링하는것을 **오케스트레이션 (Pipeline Orchestration)** 이라고 칭함.

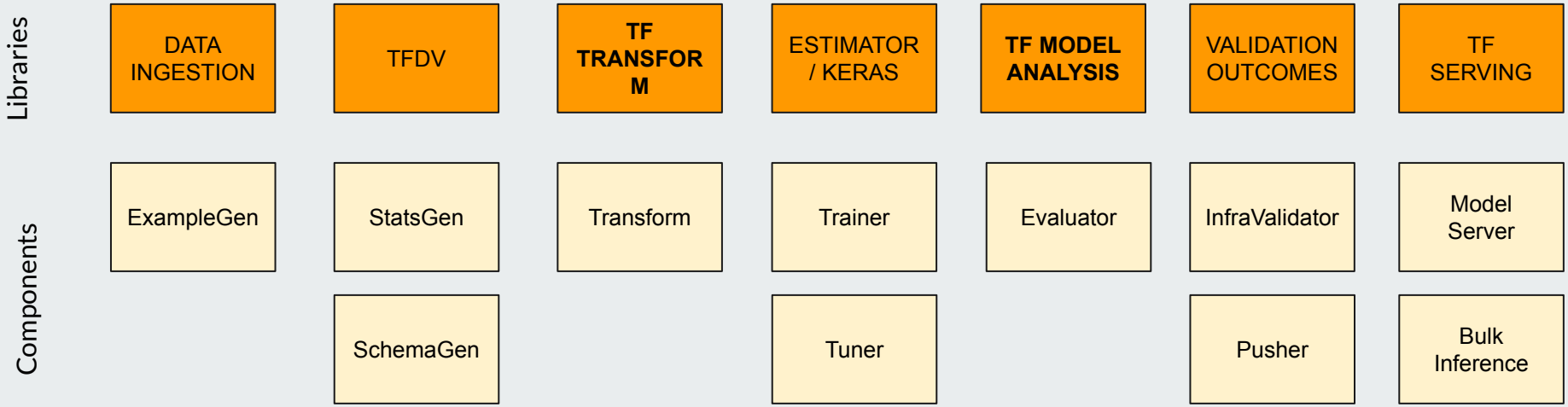
예) KubeFlow, Vertex AI, **TFX (Google)**, Michaelangelo (Uber), FBLearner (Facebook)

Pipeline Orchestration



ML Pipeline Components (TFX)





ML 모델링 vs 프로덕션 ML

	ML Modeling (Academia)	Production ML (Industry)
데이터	정적 데이터 (static data)	비정적 데이터 (dynamic data)
디자인 목표	높은 정확도	추론 퍼포먼스, interpretability
모델 학습	최적화된 모델 튜닝	지속적인 eval + 재학습

Canary Deployment Strategy (1강)

예: 공장 출하 중에 불량제품 판정하기

1. Baseline 모델만 사용

2. 새로운 모델을 **Baseline모델과 흡사한 데이터**로 테스트.

(이때 model latency 등을 확인)

3. 실제 트래픽 중, 작은 양(5%)은 ML 모델이,

나머지는 Baseline모델 사용.

4. ML 모델이 판정하는 양을 조금씩 증가해서 100%가 되었을 때 "**launched**"

Dark Launch:
실제 inference data에 대한 모델의 output을 유저에게 보이지 않고 모델 테스트 진행.

Dark launch상황에선 model accuracy를 측정하기 어려움. **(왜?)**

Canary Deployment Strategy (1강)

예: 공장 출하 중에 불량제품 판정하기

1. Baseline 모델만 사용

2. 새로운 모델을 **Baseline모델과 흡사한 데이터**로 테스트.

(이때 model latency 등을 확인)

3. 실제 트래픽 중, 작은 양(5%)은 ML 모델이,

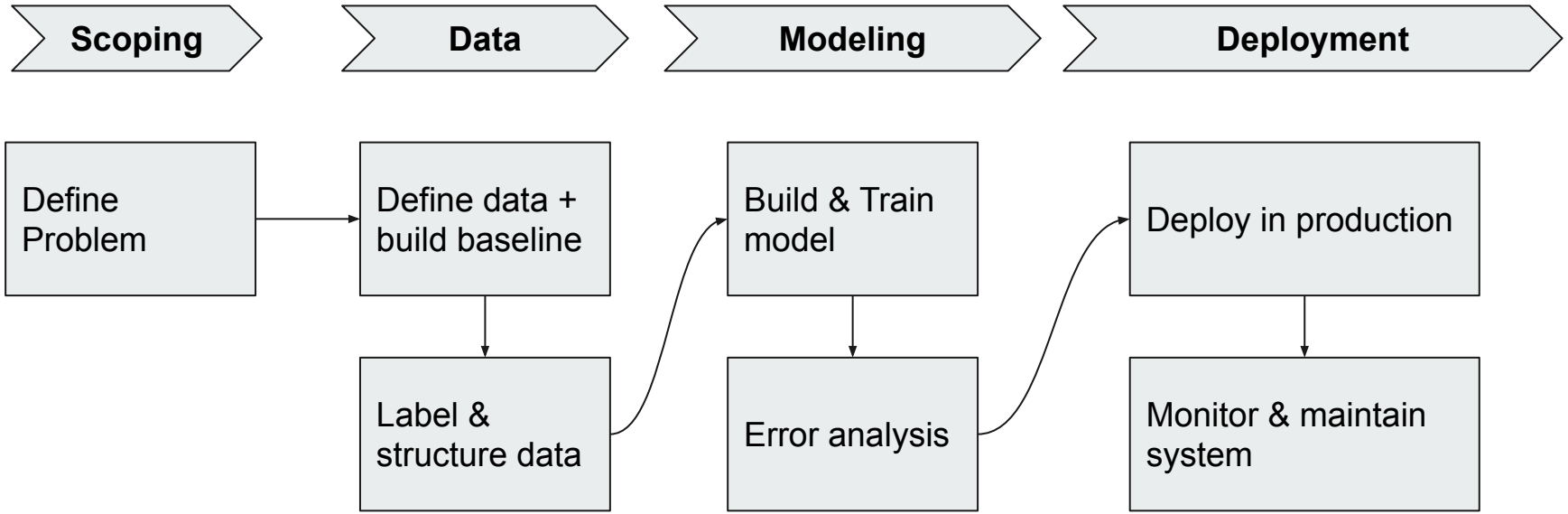
나머지는 Baseline모델 사용.

4. ML 모델이 판정하는 양을 조금씩 증가해서 100%가 되었을 때 "**launched**"

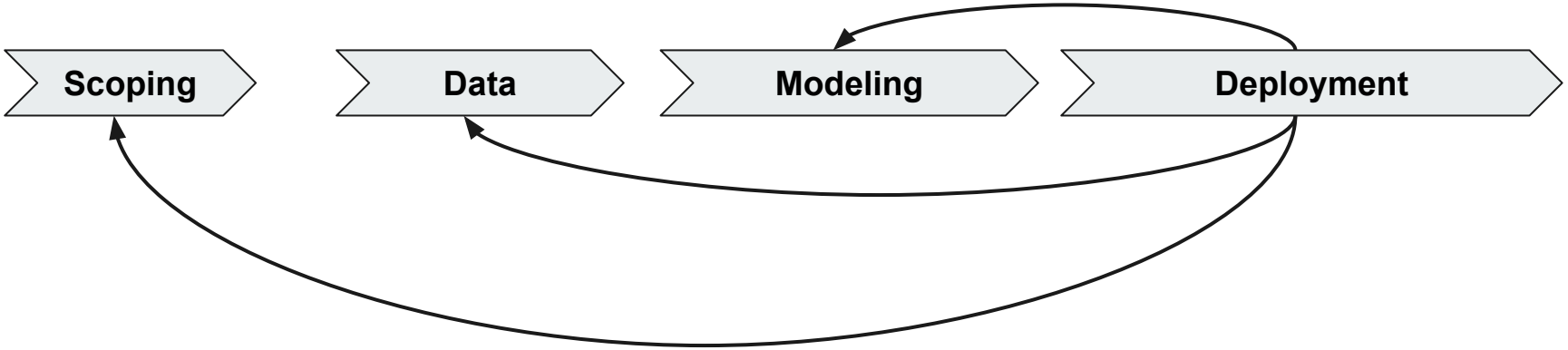
Live Experiment (LE):
실제 트래픽을 써서 실험하는 것을 칭함.

LE 실행 시엔 모델 정확도를 판명 가능.
이때부터 baseline모델과 A/B 테스팅을 할 수 있음.

Overall ML Project Lifecycle



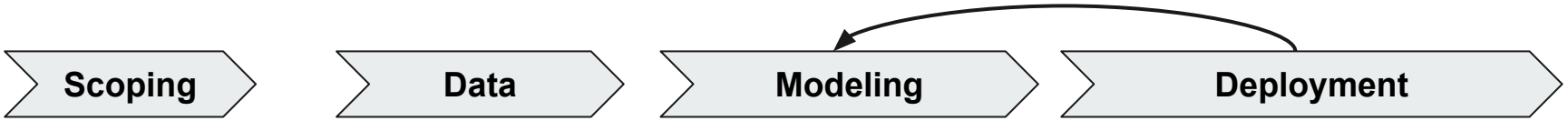
Maintaining ML System



모델 Deploy를 할 때, 때에 따라 (혹은 자주!) 이전 단계로 돌아가서 재확인 + 재학습이 필요함.

이때가 유일하게 DAG에서 싸이클이 존재.

Maintaining ML System



LE에서 모델을 deploy했는데 모델 정확도(accuracy)가 떨어진다면?

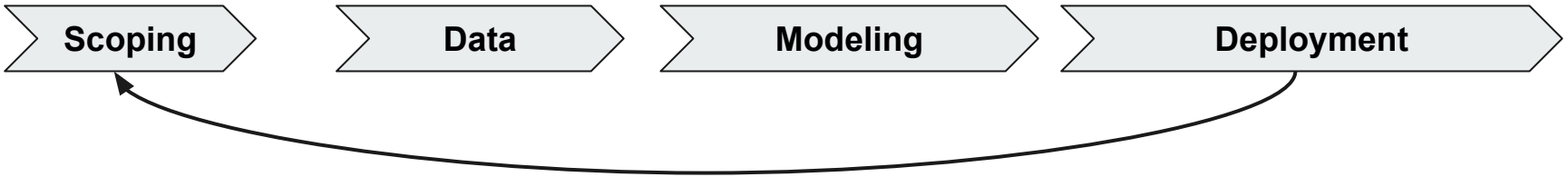
- 모델로 돌아가서 다시 모델링.
  - Feature engineering
  - 모델 아키텍처 변화
  - 하이퍼파라메터 튜닝

Maintaining ML System

혹시 baseline모델 대비 학습 데이터에 변경이 있었나?

- 데이터로 돌아가서 다시 확인.
  - Data Validation
  - Label Validation
  - **Data Drift**

Maintaining ML System



혹시 모델에 **컨셉 드리프트**가 생겼나?

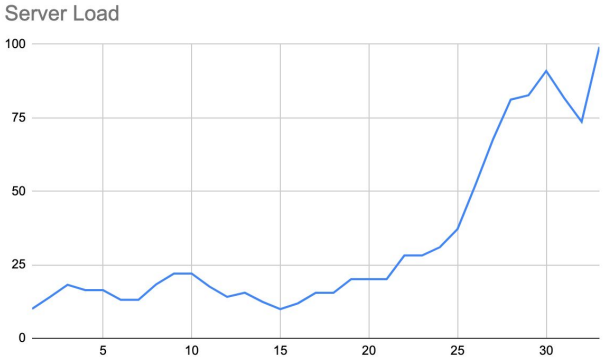
- 스코핑 / 모델 정의로 돌아가서 확인.
  - Concept Drift
  - 레이블과 Ground truth와의 비교.

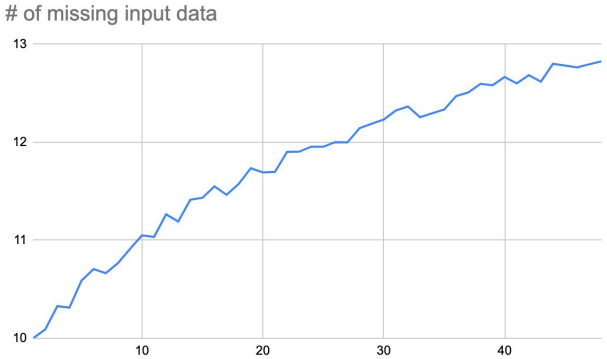
Quiz + Break Time

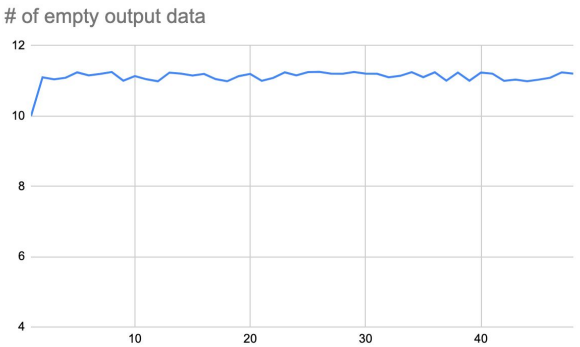
Quiz 2 Link

Metrics & Monitoring

Monitoring Dashboard







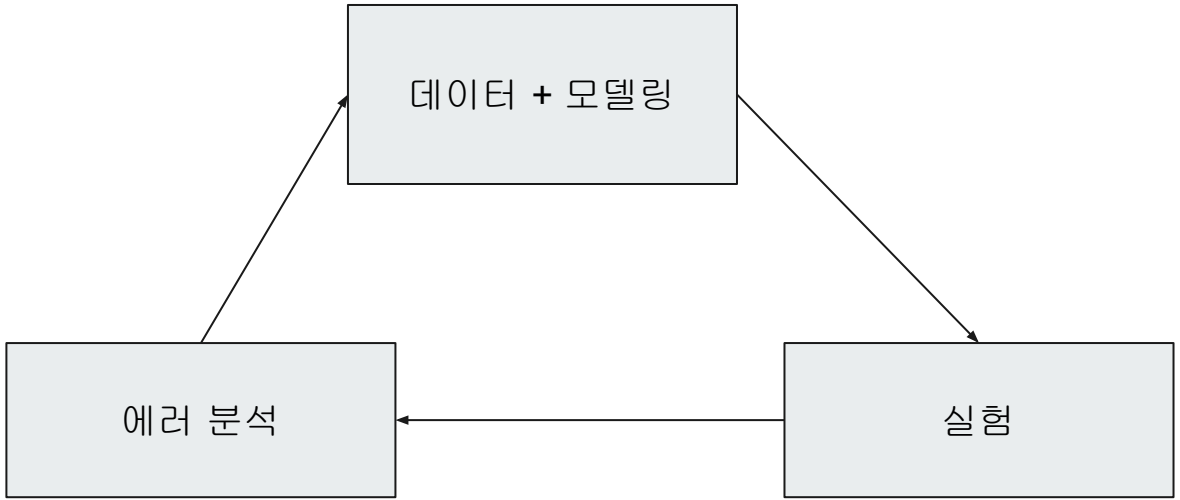
1. 가능한 시스템 에러의 요인들을 열거.
2. 각 해당하는 시스템 통계/메트릭을 정의.
3. 하나씩 모니터 대쉬보드에 트래킹 하며 모니터링 시작.

Metrics

어떤 metric을 모니터링 해야 하나?

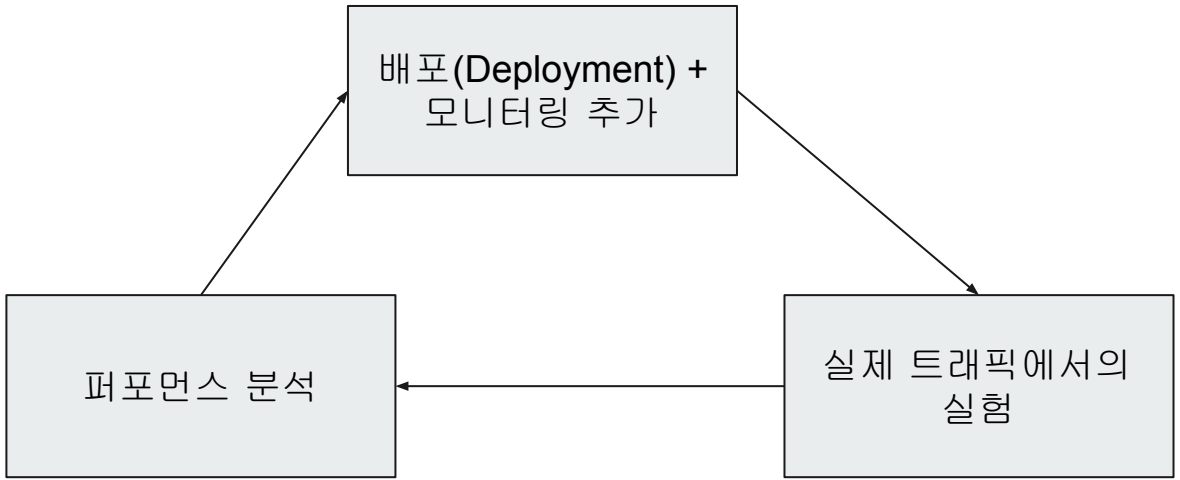
System Software Metric	• Server load • Latency • Throughput
인풋 메트릭 (Feature)	• Avg input volume • 인풋에 missing value를 가진 feature 비율 • Invalid 한 데이터 갯수
아웃풋 메트릭 (Label)	• missing output의 비율 • CTR • 학습때 썼던 Label 분포 대비 추론에 발견된 label 분포 • 사용자의 retrial / refinement

모델 생성을 위한 재반복 싸이클



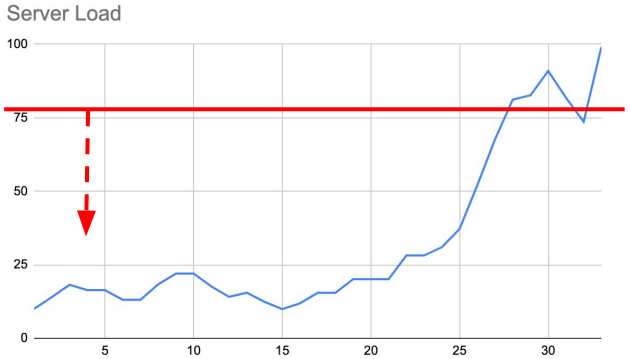
반복된 실험을 통해 필요한 데이터 + 모델링을 완성.

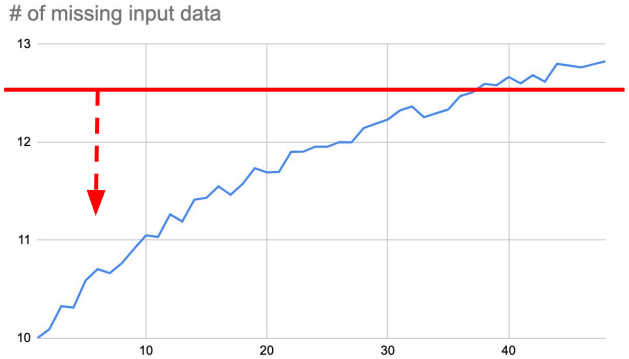
모델 배포를 위한 재반복 싸이클

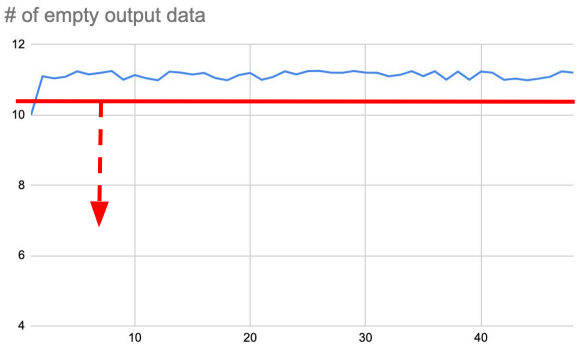


마찬가지로, 배포 (Deployment) 또한 반복적인 배포를 통해 모니터링 할 메트릭을 완성.

Threshold Setting + Alarm



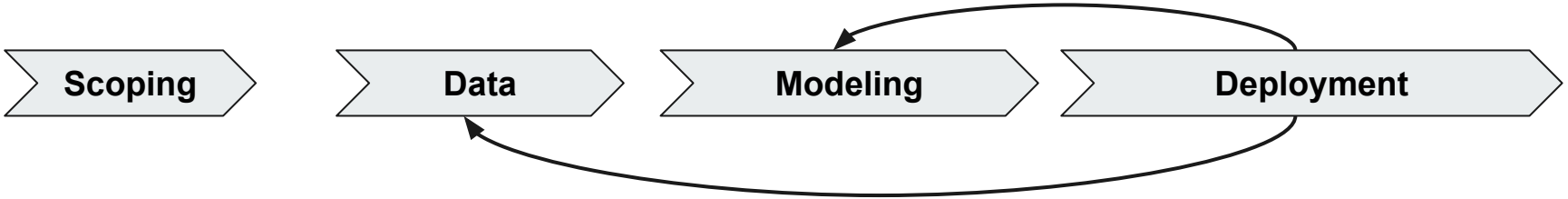




각 메트릭과 **제품 품질**을 비교하며 threshold 세팅

- 각 메트릭마다 threshold를 세팅하여 기준을 넘어설 때 alarm trigger
- ML Pipeline 및 모델을 업데이트 하며 새로운 메트릭과 Threshold도 적절하게 업데이트 필요.

Maintaining ML System

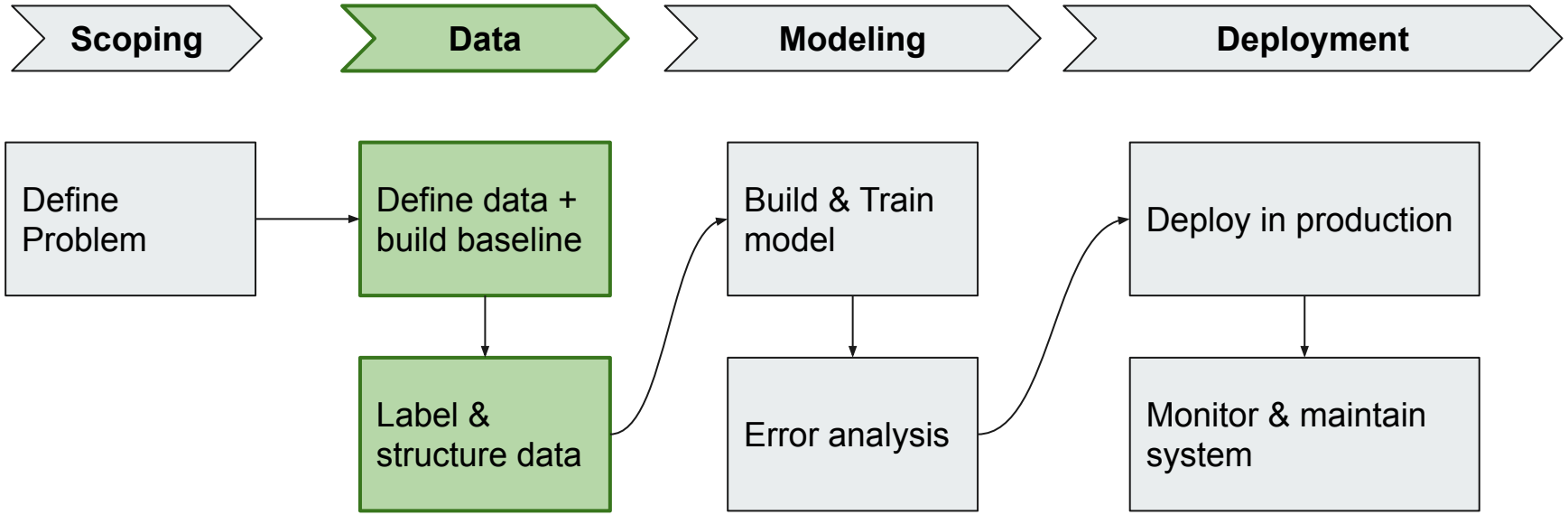


Model Update Policy: Metric Monitoring으로 에러가 발견 되었을 때 어떻게 모델 업데이트를 진행하나?

Manual update	• mission critical (의료, 자율자동) • 자주 바뀌지 않는 데이터 (자연어 처리, 사물)
Automatic update	• 자주 바뀌는 데이터 (소비 패턴, 금융, 딜리버리)

Data

Overall ML Project Lifecycle



Data Quotes

Data is the hardest part of ML and the most important piece to get it right.

(Michaelangelo, Uber)

ML with **great data** and **OK algorithm** > ML with **poor data** and **great algorithm**

(Andrew Ng)

Different Types of Data

비정형 데이터 (Unstructured data)

- 많은 양의 레이블이 없는 데이터를 수집할 수 있다.
- 인간이 수동적으로 레이블링 할 수 있다.
- 메타데이터를 추가하면서 정형 데이터를 추가할 수 있다.

예)
이미지, 음성, 텍스트

정형 데이터 (Structured data)

- 더 많은 데이터 수집이 어려울 수 있다.
- 인간이 수동적으로 레이블링 하기 어려울 수 있다.

예)
쇼핑, 날씨, 주식 가격, 부동산 가격

Data Size Matters

적은 양의 데이터 (< 10000)

- **레이블의 정확도**가 매우 중요.
- 인간이 레이블을 생성하기 쉽다.

많은 양의 데이터 (> 10000)

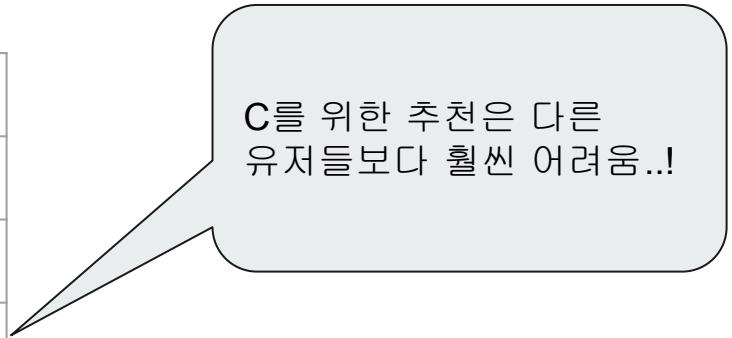
- **데이터 후처리**가 매우 중요.

빅데이터 Caveat

데이터가 많다고 무조건 모델링 문제가 쉬워지는 것은 아님.

예) 쇼핑 추천

사용자	구매히스토리 갯수
A	14,039
B	84,003
C	2



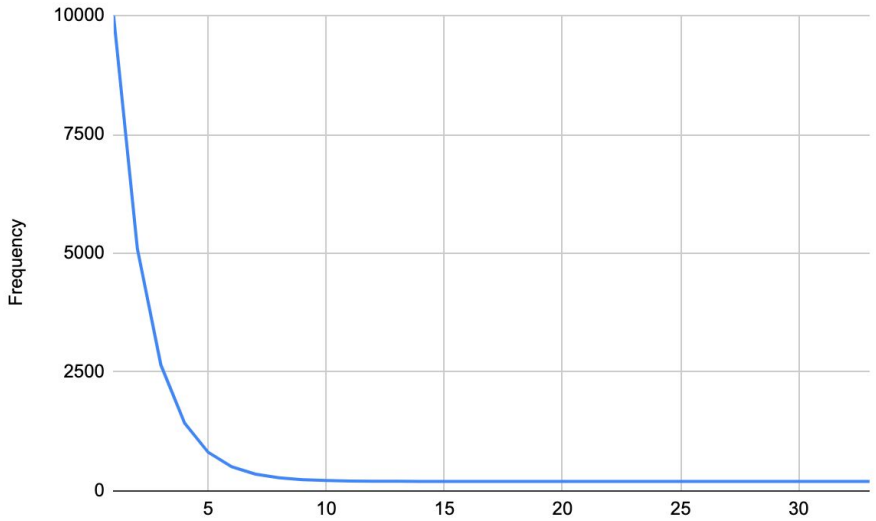
롱테일 (long tail)의 레어한 데이터포인트들이 있다면 스몰데이터 문제처럼 해결!

Long tail problems

롱테일 (long tail) 이란?
Power law처럼 길게 낮은 빈도의 데이터가 분포되어 있음.

예)

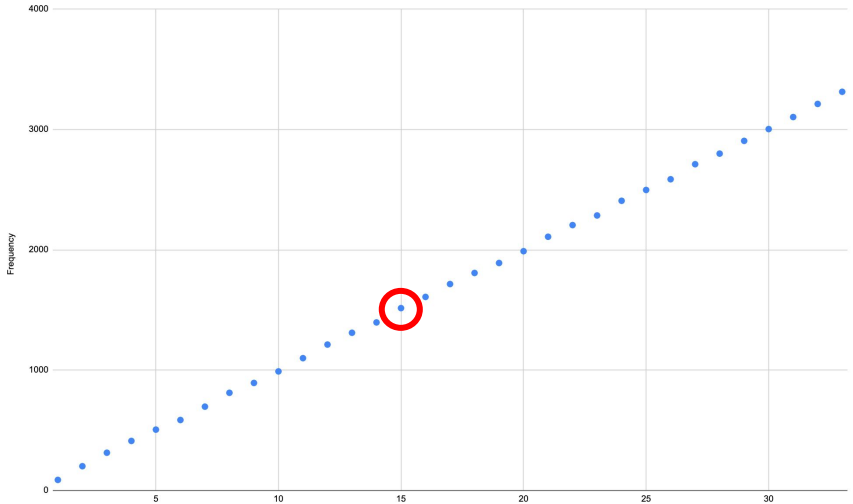
- 검색엔진에서의 검색쿼리
- 의료 예측에서 드문 환자
- 자율자동차에서 드문 사고 예방

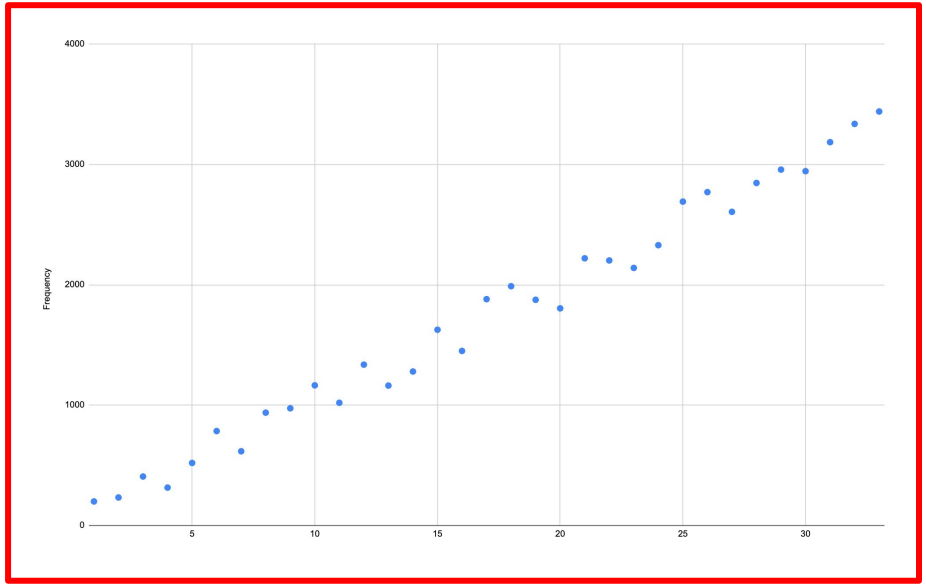


레어한 이벤트도 자주 있는 이벤트만큼 (혹은 그보다 더) 중요할 때:
- small data 문제들처럼 레이블의 정확도가 매우 중요.

좋은 데이터는 어떤 데이터인가?

- 예측 능력에 도움이 되는 데이터 (maximum predictive power)
- 도움이 되지 않는 데이터는 삭제해도 무방.





데이터 자체의 문제들

- 일관성 없는 데이터 포맷
  - “Zero”, 0, “0”, “0.0”, empty
- 학습데이터로 사용되는 데이터(Data source)에 에러가 있을 때
- Data Validation을 사용하여 데이터 소스를 모니터링 해야 함.

```
for D in data_sources:
        if not D.available: Alert()
        for d in D.column:
                clean_d = CleanData(d)
                if not clean_d.success(): Alert()
        ...
```

CleanData():
- 데이터 교정
- 데이터 분포 확인

Define Modeling Data

예) 쇼핑웹사이트의 상품 추천

모델링을 위한 데이터 (User Data)	• 상품 구매 이력 • 사용자의 demographic 정보 • 사용자의 geographic 정보
Feature Data	• 사용자 demographic: age, gender, racial_info • 거주지 정보: country, state, zip code • 검색 이력: (search query, search date) • 상품 추천 이력: (product ID, recommended date) • 상품구매 이력: (product ID, purchase date)
Label Data	• 상품 구매 이력: (product ID, purchase date) • 구매 의사 이력: (click through, date) • 비호감 의사: user clicked, but exited • 검색에서 구매까지의 # of clicks

Label vs Metric

모델 정확도가 높아도 **제품 퀄리티**와 다를 수 있음.

예) 쇼핑추천엔진의 **제품레벨 메트릭 (product level metric / topline metric)**은?

쇼핑웹사이트의 **매출 증가** → "proxy" → MAU, 비호감 의사, 구매까지의 # clicks

제품레벨 메트릭을 레이블로 사용하지 못할 경우? (e.g. lack of data)

Label: Click through → "proxy" → MAU, 비호감 의사, 구매까지의 # clicks → 쇼핑웹사이트의 **매출 증가**

Data 정의를 할때 고려해야 할 점들

- 풀고자 하는 문제 시스템 및 사용자의 데이터를 모델링에 필요한 데이터로 재해석
  - 어떤 데이터가 얼마만큼 있나?
  - 각 데이터마다 어떤 이슈가 있나?
  - **Feature:**
    - 이 데이터가 모델 예측에 도움이 될 것인가?
  - **Label:**
    - 레이블은 어떻게 정의할 것인가?
  - **Metric:**
    - 제품레벨의 메트릭은 무엇인가?
    - 다시말해, 모델의 정확도를 제품의 정확도로 재해석 할수 있나?